Application Note | Case Study | Technology Primer | **White Paper**

GigE Vision for Real-Time Machine Vision

The acceptance and use of Gigabit Ethernet technology for industrial digital imaging applications continues to grow. The increased variety and capability of cameras to leverage this powerful interface, combined with planned updates to the AIA (Automated Imaging Association) GigE Vision® interface standards makes GigE Vision based imaging more robust and more ideal than ever for real-time machine vision applications.

Real-time imaging does not always mean ultra fast acquisition. But it certainly means that images and results are available when needed. Therefore, the speed at which such a system acquires and processes data can vary significantly. But determinism must be guaranteed to ensure reliable operation of the real-time system.

The goal of this technical brief is to cover the most important aspects to consider in deploying GigE Vision cameras in a real-time machine vision system. This includes typical aspects such as latency and jitter. Other elements such as the impact of the operating system and PC, and various ways to optimize your network configurations are also presented. This should provide sufficient information to help analyze the impact of moving from a frame grabber based system to a GigE Vision solution when real-time operation is required from an area-scan camera.

GigE Vision Overview

The GigE Vision standard was first released in May 2006 by the Automated Imaging Association (AIA). Within a period of 4 years, it has become one of the dominant camera standards in machine vision. It has exceeded Camera Link in the number of units shipped in 2009, just behind Firewire-based cameras (analog cameras still representing more than 50% of units shipped).

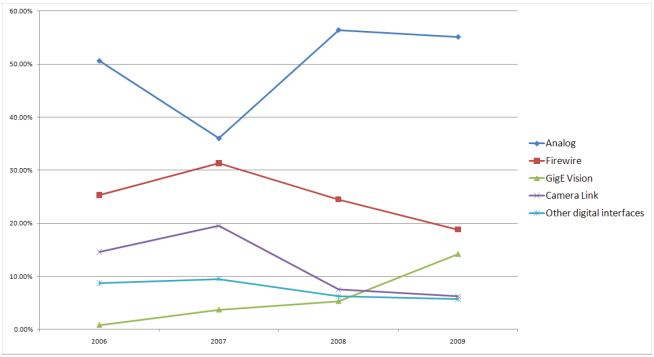


Figure 1: Evolution of percentage of cameras sold in North America (data from AIA Market Studies)

Since its inception in 2006, the GigE Vision standard has been through 2 revisions. Revision 1.1 was released in April 2009 and provided various improvements to facilitate integration into real-world applications. With release 1.2 (January 2010), GigE Vision allows non-streamable devices such as I/O boxes. This enables whole new classes of GigE Vision devices, beside cameras, to be controlled through the same software interface. This eases the integration of a variety of GigE Vision components in the machine vision system.

GigE Vision 2.0, to be released in the third quarter of 2011, focuses on improving the speed of transfer by introducing formal support for 10 GigE, IEEE802.1AX Link Aggregation, IEEE1588 Precision Time Protocol, image compression (JPEG, JPEG 2000 and H.264) and frame packing. This will offer improved opportunities for real-time machine vision systems.

continued >

Real-time Imaging with GigE Vision

Real-time applications must deal with latency and jitter. Latency is the time taken between the start and completion of a task. Jitter is the time variation when the same task is executed multiple times. When creating a real-time system, it is important to define the latency and jitter requirements and ensure the system can operate reliably and deterministically within those specifications.

In this section, we analyze the various elements affecting overall latency and jitter during a GigE Vision image acquisition. Where possible, we compare GigE Vision to Camera Link to contrast the impact of the frame grabber on the responsiveness of the real-time system.

Trigger

In most machine vision system, a trigger is used to initiate image capture. The trigger signal is typically synchronized to the object to inspect. Two basic schemes exist to forward this signal to the camera:

Hardware trigger

A hardware trigger uses a dedicated wire or electronic signal sent directly to an input pin of the camera. The latency of hardware trigger is normally extremely good, but some devices support debouncing of the trigger signal to filter off false triggers: any trigger pulses shorter than a pre-configured value are discarded. This debouncing operation adds a small latency to confirm the validity of the trigger pulse and can be of the order 1 μs or more. Some jitter can be introduced by these electronic components used to decouple the electric signal. For instance, an opto-coupler used to isolate the incoming signal can introduce different reaction time depending on the voltage difference applied at its inputs and the current provided.

It is interesting to compare a GigE Vision camera to a frame grabber for the hardware trigger situation. A Camera Link frame grabber provides an input trigger pin offering the same functionality as the one provided by a GigE Vision camera. But the frame grabber has to relay that hardware trigger signal to the camera through one of the control lines of Camera Link. This added latency is extremely low. Therefore GigE Vision and Camera Link solutions offer extremely fast responses to hardware triggers. So it is much more a question of convenience as to where it is easiest to connect the trigger cabling than a question of performance.

Software trigger

For software trigger, the application software sends a trigger command to the camera. This command is sent over the camera configuration channel, which is not as responsive as a camera input pin. In these situations, various latencies are introduced since the software typically runs on a non-real-time operating system (such as Microsoft Windows) and the creation of the command requires CPU time. Sending a software trigger is thus subject to overall availability of the CPU to process and send the command. In a well designed system, this can take a few hundredths of microseconds. But if CPU is loaded, or if other I/O activities (such as display and access to hard drive) occur simultaneously, then latency can be greatly degraded and reach milliseconds in range, especially if the I/O devices have poorly designed drivers. It is important to understand that Windows kernel driver (those that control display, I/O and the other peripherals in your PC) can easily monopolize a CPU core as they run at a very high priority. In general, software triggers are best avoided in real-time systems because of this uncertainty.

Software triggering of a GigE Vision device adds to network latency when compared to a frame grabber system. This latency can be approximated by using half of the round-trip time of a command/acknowledge packet pair on the network. This round-trip time is visible using a packet sniffer, such as WireShark (http://www.wireshark.org). For a typical GigE Vision system, this delay ranges between 100 and 500 μs, depending on the quality of the implementation of the host software and camera. In a Camera Link system, the frame grabber converts the software trigger into an electronic signal on one of the Camera Link Control lines. Hence this network latency is totally eliminated.

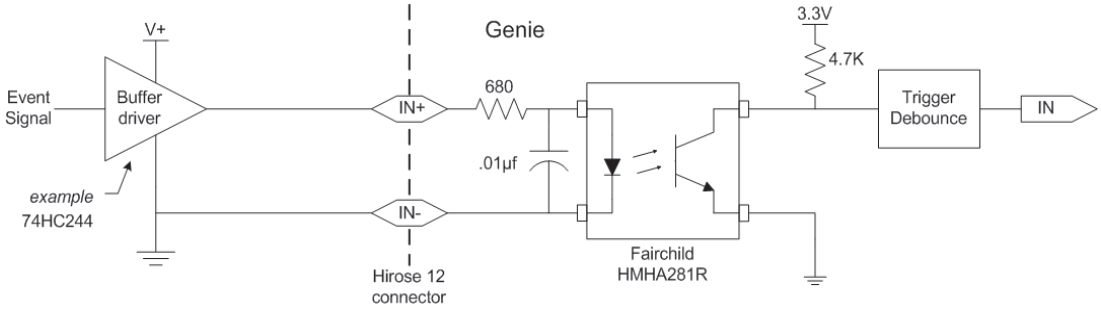


Figure 2: Example of an opto-coupled trigger pin (from DALSA Genie camera)

Synchronization using IEEE1588

GigE Vision 2.0 supports IEEE1588 Precision Time Protocol. IEEE1588 offers very precise synchronization of the internal clock running on different devices connected to the network, up to within hundredths of nanoseconds for well conceived designs.

GigE Vision devices that are IEEE1588-enabled are able to have a “common” timestamp. This facilitates implementation of multi-camera vision systems where images from different cameras must be synchronized together to a very high precision level.

GigE Vision 2.0 also extends the concept of software trigger command by allowing them to be scheduled very precisely in the “future” through the IEEE1588 common clock. This minimizes jitter at the expense of a longer latency. It also provides a simple scheme to synchronize multi-camera acquisition through a software trigger command that is broadcast on the network. This has the potential to greatly simplify I/O cabling when many cameras must be synchronized if the added latency is acceptable.

Exposure and Readout

Once the trigger signal has been processed at the camera head, the sensor can start its exposure (accumulating light into electrical charges). This section is not specific to GigE Vision, but provided for completeness as exposure and readout are significant contributors to overall latency.

Three exposure modes are typically available:

1. Camera is free-running and the trigger signal indicates to capture the next frame. This might introduce up to one frame of jitter.
2. Horizontal synchronous mode where the exposure must be aligned on the line boundary timing (i.e. HSync) of the CCD. This mode presents one line of jitter.
3. Reset mode where the sensor is reset upon trigger signal pulse arrival. Camera stays idle until the trigger is received. In this case, the jitter is minimal and in the order of magnitude of one pixel.

To minimize the jitter, it is preferable to use the reset mode if it is available on the camera.

The exposure duration is generally constant and does not introduce jitter. But it directly contributes to the latency to get the captured image in host memory.

Another big contributor to the latency is the sensor readout time. The readout time is the time it takes to read the charges accumulated in the sensor and convert them to digital information (for digital cameras). For a 60 frames per second camera, this readout operation takes 16.7 ms. Again, there is no significant jitter introduced by this operation, but it is often the largest contributor to latency in the image acquisition chain. Readout time can be shortened by reducing the area of interest to digitize, but this obviously reduces resolution. Some cameras offer high frame rates, which translate into shorter readout time.

continued >

Data Transfer

Image streaming is one aspect that directly demonstrates the capabilities of the Ethernet, upon which GigE Vision is based. The link speed for GigE Vision is generally 1 gigabit per second (though you will soon start to see 10GigE-based cameras).

When considering the data transfer time for GigE Vision, one must take into account the overhead introduced by the headers of the image packets. This is essentially the Ethernet, IP (Internet Protocol), UDP (User Datagram Protocol) and GVSP (GigE Vision Stream Protocol) headers illustrated below. Streaming packets have typically 1500 bytes, or between 8000 to 9000 bytes for jumbo packets, where the first 36 bytes after the Ethernet header are used by the IP, UDP and GVSP headers. Therefore the header/footer overhead represents about 5% of the bandwidth, not considering the data leader and data trailer packets that delineate images sent over GVSP. As a rule of thumb, one can use an overhead of 5% for standard 1500-byte packets and 1% for jumbo packets if image size are 640x480 (VGA) or larger as the data leader and data trailer have negligible impact.

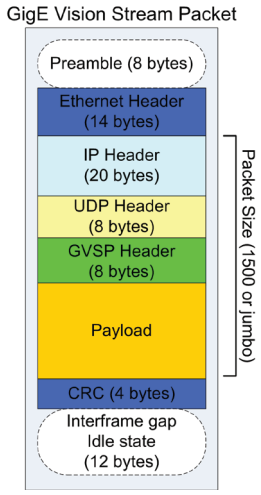


Figure 3: Ethernet Frame

To give an idea of the time is takes to transmit a packet, let's consider an 8000-byte jumbo packet. At the speed of 1 Gigabit per second, it takes 64 μs to transmit this frame over the network, plus 304 ns of preamble, Ethernet header, CRC and interframe gap.

Camera side

It is important to understand that a GigE Vision camera cannot initiate image transmission before it has sufficient data to fill at least one packet. But what interests us the most is not when it sends the first packet, but when it sends the last packet since only upon reception of the latter is the image fully available for processing. Nevertheless, ensure that the camera you select does not wait for the full image to be read out before starting transmission on the network as this would:

1. Create a large burst of traffic on the network, possibly overloading the packet storage memory of the network card, especially in multi-camera configurations;
2. Augment the latency, especially if the inter-packet delay (discussed later) has to be increased to avoid an overflow condition within the network.

Assuming that the camera sends packets as soon as they are full of data and that the acquisition rate is lower than the transmission rate, then the last packet should be fired not long after the end of readout. This added latency is not present in a frame grabber based system, but as we will see by experiment, it is really small.

PC side

For GigE Vision based systems, there is no frame grabber. It is thus the responsibility of the NIC (network interface card) combined to a special GigE Vision driver to receive the image packets and reconstruct the image in PC memory. This reconstruction involves the decoding of the GVSP packet header and the copy of the pixel data. Fortunately, today's PCs are very efficient and data copy is greatly accelerated by specialized instruction set and memory controller supporting multiple memory channels. This is an area where there is a lot of differentiation between GigE Vision software package vendors.

For a frame grabber, the story is quite different since the frame grabber has an efficient DMA (direct memory access) engine that is able to copy data directly to its final destination without having to involve the CPU. Latency and jitter are thus kept to a minimum, but are tied to the performance of the PCI Express chipset of the motherboard. It is possible to have a GigE Vision frame grabber, but this is usually not necessary at gigabit speed and does not represent a cost effective solution. It is cheaper to buy an efficient multi-core PC with a good network card that is able to sustain large bandwidth.

Once the image data is fully copied into the image buffer, the GVSP or frame grabber driver normally signals an event to indicate to the application software that the buffer is ready for processing. The latency of this signaling is generally quite fast (hundred of microseconds at most), but can quickly deteriorate into millisecond range if the CPU is heavily loaded or if other I/O activities are taking place, as explained earlier under the "Software Trigger" section.

Impact of Windows on Latency and Jitter

Microsoft Windows is the most popular operating system for PCs these days. But it is far from being a RTOS (real-time operating system). To optimize its responsiveness, it is necessary to be careful with the peripherals and drivers that are installed. Different vendors offer quite varying levels of performance in terms of CPU usage and it sometime does not take much to derail an intended real-time system running Windows.

DALSA has seen many systems where the worst-case reaction time exceeds 100 ms. These systems typically contain peripherals that have been poorly designed, many of them polling to monitor the progress of specific operations. Constant access to a hard drive swap file is a good example of an element that can greatly degrade the latency and increase the jitter. And you probably have firsthand experience with a virus scan consuming your precious CPU processing power!

Simple measures can be taken to limit the risk of undesirable drivers impacting our real-time systems: simply disable them if you don't need them! Such an example can be found by the myriad of protocols associated to a network card (see figure below). Each of these might need to inspect incoming packets and thus increase the overall latency.

This is the very reason why GigE Vision software providers offer "filter drivers" to bypass the Windows network stack. This enables efficient processing of packets, especially for streaming. The following figure shows how DALSA's Sapera LT GigE Vision software deals with GVCP (GigE Vision Control Protocol) and GVSP (GigE Vision Stream Protocol) of GigE Vision.

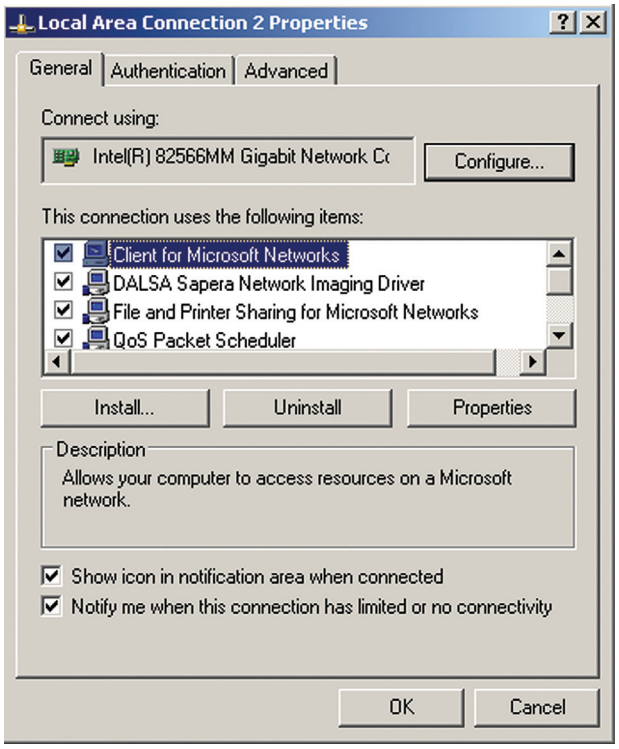
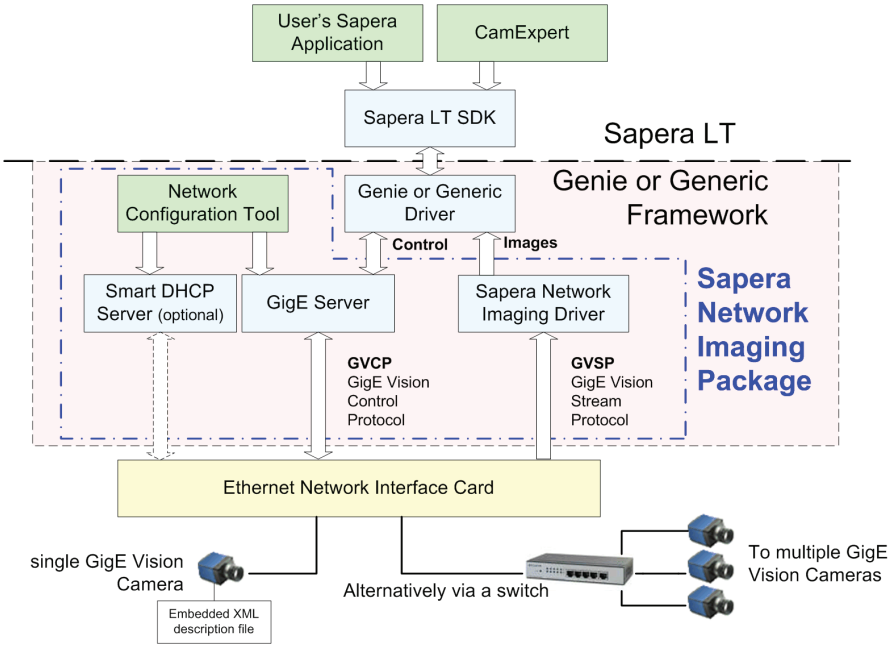


Figure 4: Example of Network Protocol that are enabled by default on Windows



continued >

Optimizing Network Settings

Interrupt Moderation

Normally, each time a packet is received by the network card, the associated driver will receive an interrupt. Obviously, when packet rate is very high (that is, at high transfer rate which is common for GigE Vision systems), this represents significant overhead. Most network cards have introduced an interrupt moderation mode where the card waits to have received a certain number of packets over a maximum period of time before issuing the interrupt. This helps reduce the burden on the CPU as it can process multiple packets during the same interruption.

The Intel Pro/1000 Network adapter provides a configuration parameter to manually adjust the NIC interrupt rate. By default the NIC driver sets this to Adaptive where the interrupt rate automatically balances packet transmission interrupts and host CPU performance. In most cases no manual optimization of the Interrupt Moderation Rate parameter is required.

In some conditions, video frames from the GigE Vision camera may be transferred to the host display or memory buffer as data bursts instead of a smooth continuous stream. The NIC may be over-moderating acquisition interrupts to avoid over-loading the host CPU with interrupts. If priority is required for acquisition transfers (i.e. a more real-time system response to the camera transfer) then the moderation rate should be reduced by manually adjusting the NIC parameter.

In the end, this is a compromise:

1. Enable interrupt moderation to minimize CPU usage, at the expense of a slight increase in latency.
2. Disable interrupt moderation to favor responsiveness of real-time system with a drawback in CPU usage.

In most situations, extra latency introduced by interrupt moderation is very low and thus the gain on CPU performance becomes more beneficial.

Packet Size

With good gigabit Ethernet connections with minimal packet resend conditions, host computer performance can generally be improved by increasing the data packet size. Each streaming video packet(s) causes an interrupt in the host computer. Therefore increasing the packet size reduces the CPU usage percentage required to handle video data from the GigE Vision camera. The argument is quite similar to the one for interrupt moderation (reduce the number of interrupts to copy a given amount of pixels).

A standard packet can have a size up to 1500 bytes. But many network cards support a jumbo packet mode that can extend that size up to 8000, 9000 bytes or even 16Kbytes. In theory, a packet could be as large as 16 KB, but the CRC (cycle redundancy check) containing the checksum of each packet is not as efficient when the packet size grows larger than 9000 bytes.

An element to consider is if the last data packet of an image is padded by the GigE Vision camera or not. The GigE Vision specification allows for this last packet to be shorter than the other packet since image size is not necessarily a multiple of the packet size. When using jumbo packets, the latency on the shorter last packet is no worse than if regular packet was used. In any case, we are talking about tens of microseconds, so not a significant impact to system latency.

Receive Buffers

Under certain conditions the host PC system CPU may be busy with tasks other than the imaging application. Incoming image packets remain in the PC memory allocated to store packets instead of immediately being copied into the image buffer. By increasing the number of NIC (network interface card) receive buffers, more incoming image packets can be stored by the NIC before it must start discarding them. This provides more time for the PC to switch tasks and move image packets to the image buffer.

Not all network boards allow increases to their receive buffer count. Among those that do, Intel NIC, different versions will have different maximum receive descriptor values.

In any case, with the amount of memory in today's PC, there is no drawback to increasing the receive buffer size to the maximum permitted by the network card. It simply provides more buffering capacity when needed.

Flow Control

The GigE Vision standard defines an inter-packet delay that can be used to manage flow control (i.e. the speed at which stream packet can be output to the network). This is useful when connecting multiple cameras to the same port of the network card, or when the network card/Ethernet switch (if used) is simply too slow to process those packets. A careful selection of equipment will ensure that the network equipment is fast enough to handle data transmitted to the wire-speed of 1 gigabit per second. Therefore, inter-packet delay is typically only used when multiple cameras are connected to the same port of the network card, through an Ethernet switch.

It is important to consider that inter-packet delay inserts a minimum delay between image packets to spread packet transmission over a longer period of time. This can directly impact system latency as more time than could be necessary is put in between those packets. The best approach for real-time imaging is to dedicate a different network port to each camera. This way, the inter-packet delay can be eliminated in many cases.

Some network equipments also supports the optional IEEE802.3 PAUSE mechanism. This is a low-level handshake to ensure the receiver of the packets is not overwhelmed by the amount of data. It can propagate a pause signal back to the transmitter, asking to momentarily stop the data transmission (with a possible impact on the overall system latency). Again, by combining network equipment that can operate at wire-speed and allocating a different network interface port for each camera in the system, we can ensure these pause requests will not be used.

continued >

Experimental Evidence

As a reality check, DALSA has performed a comparison of latency and jitter between a GigE Vision acquisition and a Camera Link acquisition. Both systems used the same PC with the same configuration. The only difference is the camera and the use of a frame grabber for Camera Link. Using an oscilloscope, we measure the time between the hardware trigger (this starts the trace capture on the oscilloscope) and the end-of-acquisition event that signals a thread to toggle a pin of the PC parallel port. So this is a setup equivalent to a real world application from the time of the trigger to the time the image processing thread can start its access to image data. The end-of-acquisition event generates the next hardware trigger to generate thousands of image acquisition and provide a significant sample size.

The GigE Vision system uses DALSA Genie HM1400 (1400 x 1024 @ 64 fps in 8-bit) running in external trigger mode. Exposure duration is set to 100 μs and the camera is operated in reset mode. We have used 1500-byte packets and 9014-byte jumbo packets to investigate effect of packet size. Inter-packet delay is set to 0. The link is running at gigabit speed. Camera is directly connected to the first port of an Intel Pro 1000PT dual port network interface card. You can get a single port version of this card for $70USD and a dual-port for $200USD. In free-running mode, the transfer rate of this camera is 93 MB/s with jumbo packets.

The Camera Link system uses DALSA Falcon 1.4M100 (which happens to have the same DALSA CMOS sensor as the Genie HM1400, 1400 x 1024 @ 102 fps in 8-bit, 80 MHz pixel clock) with the same settings as the GigE Vision system: 100 μs exposure duration using sensor reset mode. The camera is connected through a DALSA Xcelera-CL PX4 PCIe frame grabber. This frame grabber uses a 4-lane PCI Express interface providing 1 GBytes/sec of transfer to the PC.

The PC is based on a SuperMicro X8STE motherboard. It hosts a quad-core Intel Core i7-860 running at 2.8 GHz. This provides 8 processing units when Hyperthreading is considered. Two DDR3 channels clocked at 1333 MHz (3 GB of memory in total) complete the picture. Tests are performed under Windows XP.

Figure 5: GigE Vision System: Genie HM1400 and Intel Pro 1000 PT dual port

Figure 6: Camera Link System: Falcon 1.4M100 and Xcelera-CL PX4 dual

DALSA Genie HM1400	www.dalsa.com/prot/mv/datasheets/genie_hm1400_xdr.pdf
Intel Pro 1000 PT dual port	ww.intel.com/Assets/PDF/prodbrief/pro1000_pt_dualport_server_adapter.pdf
DALSA Falcon 1.4M100	www.dalsa.com/prot/mv/datasheets/03-070-20034-00_Falcon_HG.pdf
DALSA Xcelera-CL PX4 dual	www.dalsa.com/prot/mv/datasheets/xcelera_cl_px4_dual_072408.pdf
SuperMicro X8STE motherboard	www.supermicro.com/products/motherboard/Xeon3000/X58/X8STE.cfm
Intel i7-860 quad-core	www.ark.intel.com/Product.aspx?id=41316f

Table 2: Datasheets of Equipment for the Experiment

Network Settings of GigE Vision System

The following figure presents the network settings used for jumbo packet transmission over the Intel Pro 1000 network card. We increased the number of receive descriptors to the maximum allowed and enabled jumbo packets of 9014 bytes.

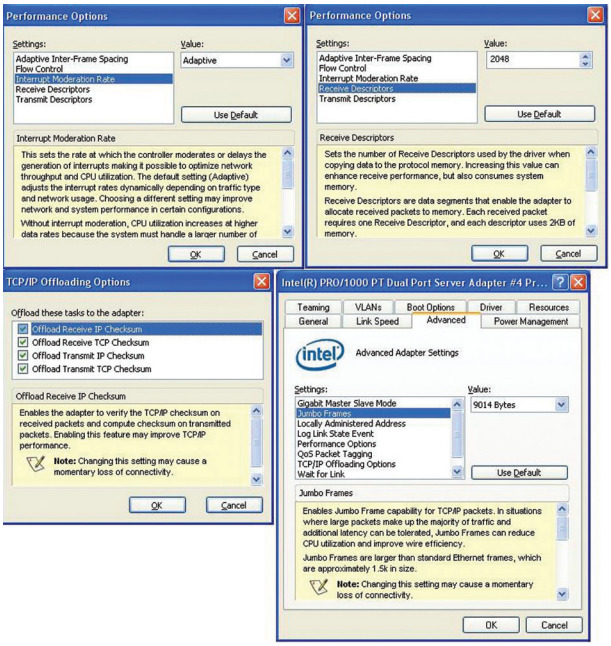


Figure 7: Network Card Settings

GigE Vision with no load on CPU

This first test looks at CPU usage when no memory copy or processing load is put on the CPU. This provides a baseline reference for the rest of the tests. Using 1500-byte packets, Windows Task Manager reports a 4% CPU load for data transfer at 92 MB/s. By increasing to 9014-byte jumbo packets, CPU usage is reduced by half to 2%.

Loading the CPU

To get meaningful worst-case results, the evaluation of latency and jitter must be performed on a system that is running near its capacity. To achieve that, the experiment runs two tools that load the CPU as much as possible (next to 100%):

1. Multiple threads performing memory copy operations
2. Benchmark application performing processing

Memory-bound operations

To load the CPU memory controller, the experiment runs multiple instances of a homemade infinite memory copy tool. As you can see, CPU usage increases dramatically. It must be noted that no images are lost during the GigE Vision system acquisition since the GigE Vision acquisition driver runs in Windows kernel at a higher priority level than the memory copy tool. Nevertheless, loading CPU memory controller provides a more difficult scenario to the memory controller.

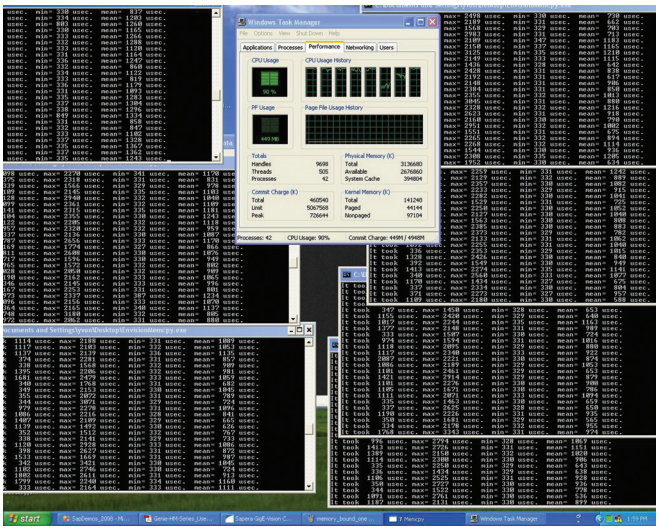


Figure 9: Memory-bound Scenario

Processing-bound operations

In order to stress the CPU processing unit, we run SiSoftware Sandra benchmarks, as shown below. This demonstrates the relative strength of this machine compared to an Intel Core 2 quad processor of the previous generation.

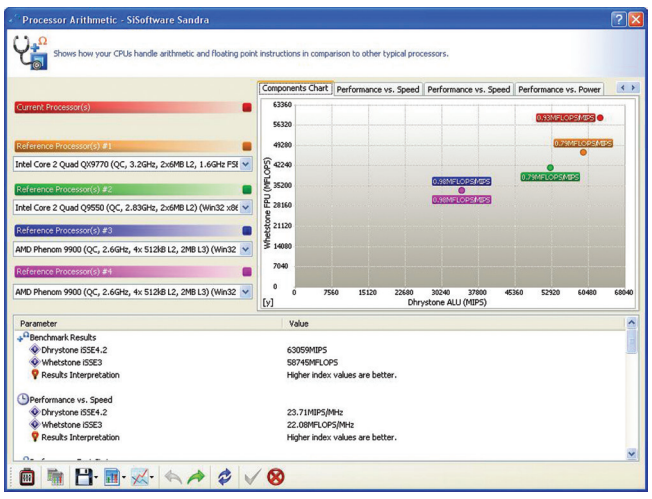


Figure 10: Processing-bound Scenario

continued >

Experimental Measurements

Having the memory copy and processing tools running on the machine, we start the experiment with various configurations:

1. GigE Vision system with 1500-byte packets
2. GigE Vision system with 9014-byte packets
3. Camera Link system

The oscilloscope is activated by the hardware trigger signal sent to the GigE Vision camera or Camera Link frame grabber and it receives a pulse from the PC during the end-of-acquisition event. This event is signaled once the full image buffer is ready for processing. The associated callback sends the pulse on the parallel port of the PC, which is attached to the second channel of the oscilloscope.

The following table summarizes the jitter results obtained. We differentiate between the typical jitter were most of the end-of-transfer events occur, as opposed to the worst-case jitter obtained from the most delayed end-of-transfer during the 1 hour test. As reference, the table also lists the jitter as a percentage of the camera readout time (15.6 ms for GigE Vision, 10 ms for Camera Link).

The figures below show the actual oscilloscope screen shot for each scenario. Each of them accumulated data for about 1 hour, running near the nominal speed of the camera (1400 x 1024 @ 64 fps for the GigE Vision system, 1400 x 1024 @ 100 fps for the Camera Link system) on a fully loaded CPU running both memory copy and processing benchmarks.

The first yellow bar at the top shows the worst-case jitter. The second yellow bar is the typical jitter. Each square on the display represents 500 μs. The latency from hardware trigger to the end-of-transfer event is extremely close to “1/frame rate” in all cases.

The dense cloud of blue dots on the left represents the typical jitter area, where most of the end-of-transfer pulses are registered.

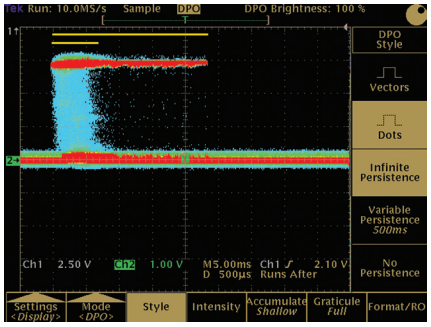


Figure 11: GigE Vision system latency measurements (1500-byte packets)

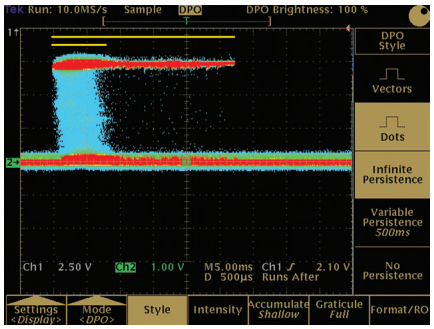


Figure 12: GigE Vision system latency measurements (9014-byte packets)

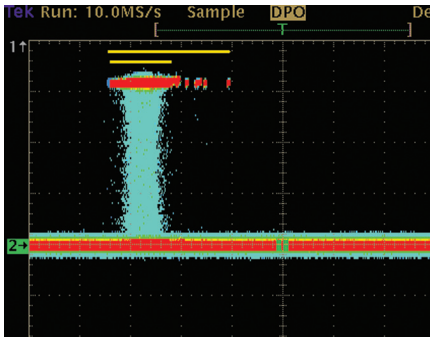


Figure 13: Camera Link system latency measurements

	GigE Vision (1500 bytes)	GigE Vision (9014 bytes)	Camera Link
Typical jitter	0.7 ms (4.5%)	0.8 ms (5.1%)	0.6 ms (6%)
Worst-case jitter	2.4 ms (15.4%)	2.7 ms (17.3%)	1.2 ms (12%)

Table 3: Experimental Jitter Comparison (fully loaded CPU)

Conclusions

A number of important things can be deduced from this experiment:

1. It is true that using a frame grabber will lead to less jitter than a GigE Vision system. The frame grabber provides electronic circuitry dedicated to data acquisition and transfer whereas the CPU is shared across all processes running on Windows. The additional amount of jitter introduced by the network connection and GigE Vision imaging driver over a Camera Link system is no more than 1.5 ms in these tests.
2. For the GigE Vision system, the worst-case jitter experienced is 2.7 ms with CPU fully loaded. This is less than what can be tolerated in many real-time machine vision systems where the data transfer time of a single image is 10 ms or more (equivalent to a 100 fps or lower frame rate area-scan camera).
3. For a GigE Vision system, jumbo packets slightly increase the typical and worst-case jitter, but drastically reduce CPU usage. Using jumbo packets is usually a good compromise.
4. The latency from trigger to end-of-acquisition is essentially the readout time. Jitter becomes much more disruptive to real-time performance than the added network latency, which is essentially the last packet transmission and its copy to the image buffer on the PC. This is true if the network bandwidth (1 gigabit per second) exceeds the acquisition bandwidth of the camera. Otherwise, image data will accumulate in the camera and the latency will start to increase. This scenario must be avoided for real-time system whenever possible.
5. It is important to use “good” components (i.e. PC, network card, CPU, …) to achieve these kinds of results. These “good” components can be found off-the-shelf and are not overly expensive. They can sustain wire-speed (i.e. 1 gigabit per second). At the time of writing, Intel i7 CPU family is particularly well suited to good performances as the CPU takes on the role of the frame grabber in moving data into the image buffers. Intel Pro 1000 network card family offers excellent performance for the price.

In order to get good real time performance with GigE Vision, here are a few additional recommendations:

1. In a multi-camera system, connect only one camera per network card port if possible. This limits the contention for network bandwidth and ensures the overall latency is minimized and more or less constant.
2. If you need to insert an Ethernet switch between the cameras and the network card, ensure it has enough buffering capacity to cope with the incoming bandwidth from the cameras. A good Ethernet switch should insert less than 100 μs of latency and minimal jitter. And don’t use an Ethernet Hub!
3. Ensure the camera acquisition speed is slower than the network speed. This is to avoid accumulation of image packets in the camera on-board memory which will directly translate into increased latency.

The above measurements have us conclude that GigE Vision provides a performance level suitable to many real-time machine vision systems. This corroborates the trend in market data that GigE Vision is outpacing the growth rate of all other machine vision digital camera interfaces.

continued >

www.dalsa.com

Americas
Boston, USA
Tel: +1 978-670-2000
sales.americas@dalsa.com

Europe
Munich, Germany
Tel: +49 8142-46770
sales.europe@dalsa.com

Asia Pacific
Tokyo, Japan
+81 3-5960-6353
sales.asia@dalsa.com

DALSA is an international leader in digital imaging and semiconductors and has its corporate offices in Waterloo, Ontario, Canada.